Adversarial Generation and Evaluation of Image Captions

Aishwarya Rajan ar6382@nyu.edu
Kawshik Kannan kk4161@nyu.edu
Prasanthi Gurumurthy pg1899@nyu.edu
Sahana Upadhya su575@nyu.edu

Abstract

GANs were originally designed to generate continuous data and have become very successful in the field of Computer Vision. Following recent success, we use the Gumbel Softmax relaxation technique to overcome the non-differentiability issue when dealing with discrete data. Using this model, the generator can learn to generate realistic sentences based on feedback from the discriminator as it learns important, discriminative features for telling them apart. We also correlate the discriminator scores with human evaluation scores to understand how close their behaviours are, and propose to standardize the use of a discriminator as a evaluation metric. We focus on image captioning and show major pitfalls and workarounds by exploring the capabilities of the model in conditional and unconditional generation settings. We are the first few people who have tried to generate sentences conditioned on a given context (images) in an adversarial setting using a reinforcement learning free method.

1 Introduction

GANs (Goodfellow et al., 2014) consist of a generator that learns to produce realistic samples and a discriminator that tries to distinguish between a real and a fake sample. These models have been very successful in generating continuous data such as images. In recent times, GANs have been extended to the generation of discrete data as well (Nie et al., 2018; Kusner and Hernández-Lobato, 2016).

Current models for natural language generation (NLG) tasks are trained with standard cross-entropy based losses which implicitly try to minimize the KL divergence which in turn optimizes for the mean of the original distribution as shown in Fig 2. In the context of NLG tasks, this could mean that the model does not generate meaningful sentences even though the sentences have a low Maximum Likelihood loss. This problem can be addressed by the use of Generative Adversarial Networks (GANs) that exhibit a mode-seeking behaviour since they optimize the Jensen-Shannon divergence instead. In other words, the model tries to mimic one mode of the original distribution with a high probability and a wide support as in Fig.3.

Current evaluation metrics for NLG tasks rely on word-overlap based metrics such as BLEU and ROUGE which do not capture some important factors used by human evaluators. Under the GAN setting, the discriminator can also be viewed as a dynamically learnt evaluation metric for tuning the generator. Once trained, the discriminator would have learned important features for identifying a good sentence and hence can be used as an evaluator.

Our contributions in this work include the use of a GAN for caption generation conditioned on images from the COCO dataset (Lin et al., 2014). The conditional generation of captions adds suitable constraints which make the problem harder owing the necessity of understanding the contents of the image. The fact that the same image can be described in multiple ways adds to the uncertainty of generation in the conditional setting. However, once this relationship is modelled, the generation process becomes more simpler when compared to unconditional generation.

2 Related Work

GANs (Goodfellow et al., 2014) were originally introduced for continuous spaces and extending their use for discrete spaces has been an active research topic for the past few years. Previous works have dealt with this issue by either using

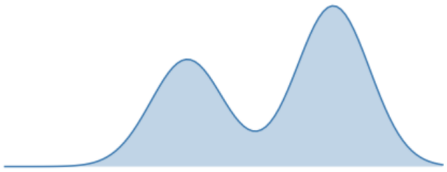

Figure 1: Data Distribution

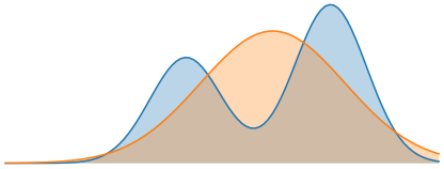

Figure 2: Mean Seeking Model Distribution

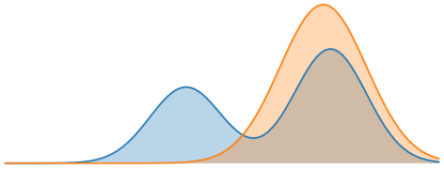

Figure 3: Mode Seeking Model Distribution

Reinforcement Learning, reformulating the problem to continuous space or approximating the discrete data.

Reinforcement learning has been largely used in GANs before. SeqGAN (Yu et al., 2017) treats the generation task as a sequential decision making process and uses policy gradients to update the generator directly. MaliGANs (Che et al., 2017) use an objective based on importance sampling to bring the objective closer to Maximum Likelihood based training and reduce variance in the updates to the generator. RankGAN (Lin et al., 2017) uses a ranking objective instead of the standard binary cross entropy loss and updates the generator using policy gradients. LeakGAN (Guo et al., 2017) proposes to leak information from the discriminator to the generator during the generation process to guide the generator more frequently and informatively. MaskGAN (Fedus et al., 2018) fills in missing tokens conditioned on surrounding text using policy gradients with a seq2seq model.

TextGAN (Zhang et al., 2017) uses a kernel based moment matching mechanism via a kernelized discrepancy matrix instead of the standard adversarial loss to force the distributions of the generated and the real sentences to be close while also reconstructing the latent features for guiding it better. FM-GAN (Chen et al., 2018) proposes to use optimal transport to match latent feature distributions of real and fake sentences. These methods opt the soft-argmax operator to approximate the argmax operator in the generator. Kusner Hernandez-Lobato (Kusner and Hernández-Lobato, 2016) provide some initial experiments for training GANs using the Gumbel Softmax relaxation trick which defines a continuous distribution over the simplex that can approximate samples from a categorical distribution ((Jang et al., 2016); (Maddison et al., 2016)). RelGANs is a recent technique that uses the Gumbel softmax relaxation technique to deal with the non-differentiability of the sampling operation in the generator and uses temperature to control the tradeoff between sample quality and sample diversity. Our work is closely related to RelGAN and explores this method in detail to find pitfalls and propose new novel objectives to overcome them.

3 Method

In this project, our objective is to train a GAN to generate captions given an image (Figure 4).

3.1 Generative Adversarial Networks

Generative models are modelled as learning an approximate distribution $q(\mathbf{x})$ of a set of i.i.d data points $(x_1, x_2, ..x_n)$ drawn from an original distribution $p(\mathbf{x})$. To generate realistic points, the network is trained in an adversarial fashion by first learning a generator G that transforms samples from a known distribution like uniform/gaussian(say, $\mathbf{z}$) into samples that approximate those drawn from $p(\mathbf{x})$. To learn G, a discriminator network is trained to take samples $G(\mathbf{z})$ and real inputs $\mathbf{x}$ and predict the probability that the input is actually drawn from the real distribution $p(\mathbf{x})$. The objective is now to obtain an equilibrium of generator and discriminator networks.

$$\mathcal{L}_{GAN} = E_{x\sim p_x} \log D(x) + E_{z\sim p_z} \log[1 - D(G(z))]$$

$\mathcal{L}_{GAN}$ is optimized by maximising it wrt D and minimizing wrt G.

3.2 Generator models

Popular text generative GANs like (Yu et al., 2017) use an LSTM backbone as a generator.(Nie et al.,

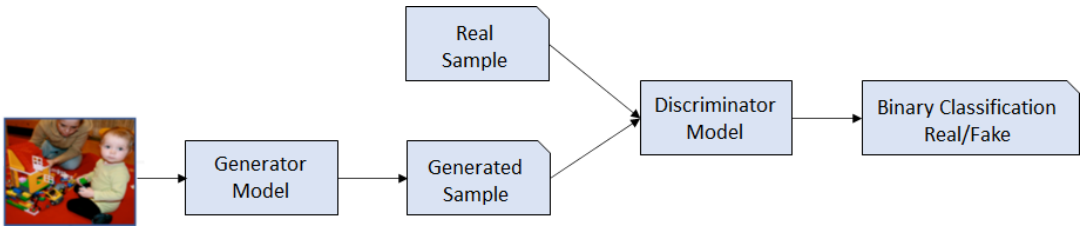

Figure 4: Architecture Diagram

2018) introduces the concept of relational memory used as generator module. LSTM based generators might be a bottleneck as it packs all information about the previous text sequences into a common hidden vector. Moreover LSTM based generators are more prone to mode collapses as it may not be expressive enough to represent all the modes of the data distribution,

We experimented with two different generator backbones - LSTM and Transformers.

3.2.1 LSTM

LSTM generates words in the sentence sequentially given the start token as the input. At every time step t, the LSTM takes the previously generated token and the previous hidden state as input and generates the new hidden state. This continues until the end symbol is generated.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = o_t \circ \sigma_h(c_t)$$

where x_t is input vector, f_t is forget gate's activation vector, i_t is input/update gate's activation vector, o_t is output gate's activation vector, h_t is hidden state vector , $\tilde{c}_t$ is cell input activation vector, c_t is cell state vector W, U, b are weight matrices and bias vector parameters.

3.2.2 Transformers

Transformers consist of an encoder-decoder architecture composed of modules such as multi-head attention and feed-forward layers stacked on top of each other multiple times. The input and target sentences are embedded into an n-dimensional space. Positional embeddings are added to these embeddings during self attention in order to give every word a relative position in the sequence. Attention in general is computed as described below, where Q, K, V are matrices of queries, keys and values respectively.

$$\text{Attention(Q, K, V)} = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V$$

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions (Vaswani et al., 2017).

$$\text{MultiHead(Q, K, V)} = \text{Concat(head}_1\text{,.. head}_h\text{)W}^O$$

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in R^{d_{\text{model}} \times d_k}$, $W_i^K \in R^{d_{\text{model}} \times d_k}$, $W_i^V \in R^{d_{\text{model}} \times d_v}$, $W^O \in R^{hd_v \times d_{\text{model}}}$

3.2.3 Gumbel Softmax

In discrete data generation, the gradients of the generator loss cannot backpropagate to the generator via the discriminator. In order to deal with this issue, we apply the Gumbel-Softmax relaxation technique which defines a continuous distribution that can approximate samples from a categorical distribution (Jang et al., 2016; Maddison et al., 2016). Under this setting, sampling follows the distribution paremeterized by temperature and G in the below equation where G is a Gumbel distribution and U is a Uniform distribution. The temperature parameter is used to balance the sample quality and sample diversity. Larger temperature encourages exploration for better sample diversity while smaller temperature encourages exploitation for better sample quality.

$$y_{t+1} = \text{softmax}(1/\text{temperature} * (\text{logits} + \text{G}))$$

$$y = \text{softmax}(\beta * (\hat{\text{y}} + \text{G}))$$

$$G = -\log(-\log \text{U}))$$

3.2.4 Autoregressive Discriminator

PatchGAN is a type of discriminator for generative adversarial networks which only penalizes structure at the scale of local image patches. Such a discriminator effectively models the image as a Markov random field, assuming independence between pixels separated by more than a patch diameter. Similarly one way to introduce a similar concept for text is to produce scores for sub-sequences. Such a model can help the generator pinpoint where exactly it could have gone wrong during the generation process. We believe this objective would help overcome the instabilities during training.

4 Experiments and Results

4.1 Dataset

The MS COCO Image captions dataset (Lin et al., 2014) consists of 124k images with 5 captions per image with 83k images for training and 41k for validation resulting in 420k training instances and 205k instances for validation. We opted to use a part of the dataset to conduct our experiments since this is a very large dataset. Particularly, we use about 10% of the images with all 5 captions resulting in 8k images and 40k instances for training, 4k images and 20k instances for validation. This was done to deal with the time constraints and the computational resources available for training. We instead conduct experiments focused on the usability of the proposed method exploring where it succeeds and fails and conclude that a similar behaviour is to be expected when trained on the entire dataset.

4.2 Model Specifications

We opted to use a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) for the generator as a baseline model and further build on this by using bidirectional LSTMs (Schuster and Paliwal, 1997) and stacked LSTMs (Yao et al., 2015). LSTMs have been used in previous work ((Yu et al., 2017)) as generators and achieved good results in text generation. Specifically we explore the impact of embedding dimension, hidden dimension and different variations of LSTMs (stacked, bidirectional).

Nowadays, with the huge success of transformers (Vaswani et al., 2017)in NLP, they have been successfully adopted as standard models for all kinds of tasks. We implement a transformer network with positional embeddings used only during self attention for use as a generator and a discriminator. The generator generates text in an autoregressive fashion during adversarial training while it decodes in parallel during Maximum Likelihood based pre-training. Attention masks for padding were used to focus only on the relevant parts of the sentence. Specifically we opt for a hidden size of 256, embedding size of 256, 4 layers, 8 heads for multi-headed attention in case of the generator. Whereas, we use a embedding size of 64, a hidden size of 128, 8 heads and 4 layers for the discriminator.

Convolutional Neural Networks have been commonly used as discriminators (Yang et al., 2017) for GANs in NLP (Kim, 2014). It employs a convolutional layer with variable sized filters to extract relationships of different word lengths and a max pooling layer over the entire input sentence for each feature map as shown in Figure 6.

RelGAN (Nie et al., 2018) proposes a variation of this architecture (shown in Figure 7 by using multiple embeddings to simulate the behaviour of multiple discriminators to give different perspectives and extract diverse and important relationships between generated and real sentences for discrimination. We employ this model as our baseline discriminator with embedding size 64 and filter sizes 2,3,4 with 300 filters each.

We experiment with conditional and unconditional generation for training the GAN. We use a Resnet18 model initialized with weights pretrained on Imagenet as our backbone in all our conditional generation experiments. During conditional generation, we have two settings where we either fix the weights of the image backbone and one where we train it together with the text generator. In case of the LSTM based models, the first input is the flattened representations of the image and this is used as context for caption generation. The conditional transformer model instead uses the convolutional feature maps (H x W x C) directly after flattening across the spatial dimensions (HW x C) in a cross-attention fashion to decode the captions.

The temperature β follows an exponential increase with respect to training steps and reaches the maximum temperature (100) at the final step.

$$\beta_n = \beta_{max}^{n/N}$$

where β_n represents the temperature at the n^{th} step, β_{max} denotes the maximum temperature, n denotes the current step and N denotes the maximum num-

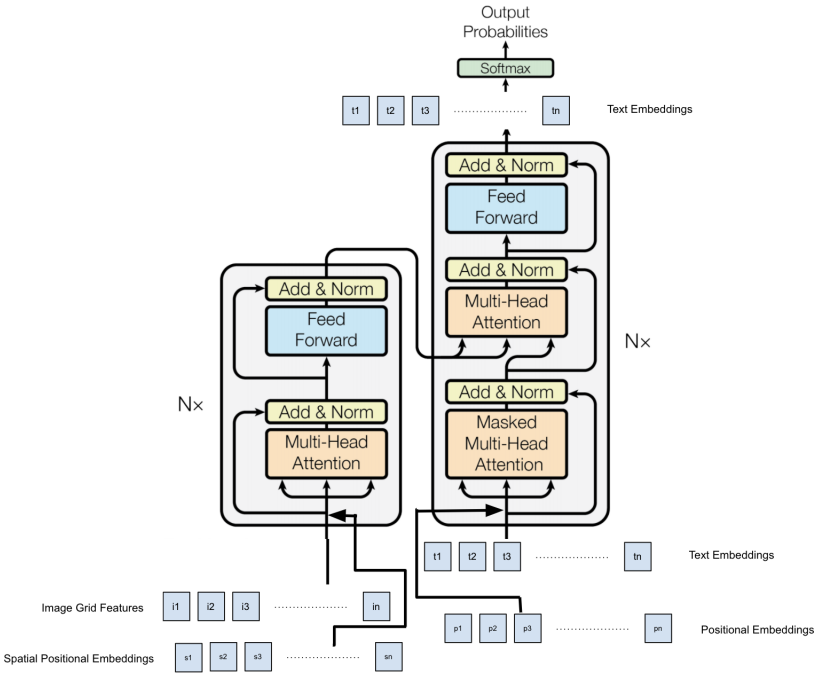

Figure 5: Transformer

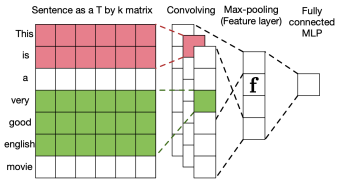

Figure 6: CNN Discriminator (Chen et al., 2018)

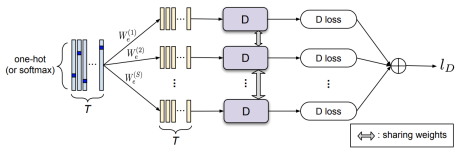

Figure 7: Discriminator proposed in RelGAN (Nie et al., 2018)

ber of states. Unless explicitly stated, a maximum temperature of 100 was consistently used in all our experiments.

4.3 Results and Analysis

Previous work commonly report two different metrics for comparison, Negative Log likelihoods for real data and BLEU scores. We present these metrics for different variations of our model for comparison.

Model	NLL_{gen}	BLEU-4
MLE	0.718	0.305
SeqGAN	1.082	0.294
RankGAN	1.344	0.264
LeakGAN	0.679	0.355
pretrained RelGAN(100)	0.756	0.502
pretrained LSTM	2.07	0.008
pretrained Transformer	1.01	0.012
RelGAN(100)	2.15	0.141
LSTM from scratch	4.23	1e-4
Transformer from scratch	3.27	1e-4

Table 1: NLL and BLEU score comparison of text generation models. (Our models are highlighted)

$$NLL_{gen} = -E_{r_{1:T} \sim P_r} \log P_\theta(r_1, r_2, \ldots, r_T)$$

Our models were incredibly unstable during training. The failure models for GANs were more prominent when generating text, with mode collapse being the frequent trouble maker. The authors of RelGAN use temperature in the gumbel softmax relaxation and pretraining to handle these failures. However they train for just 5000 iterations after pretraining and claim to get better results after this stage. We believe that this stage is primarily used for learning to generate diverse sample owing to these exponential increase in temperature

No of Steps	NLL_{gen}	BLEU-4
1 gen step	2.43	0.001
2 gen steps	1.01	0.012
5 gen steps	1.86	1e-4

Table 2: NLL and BLEU scores for different generator:discriminator step ratio when trained with a CNN discriminator

Temperature	NLL_{gen}	BLEU-4
1	22.85	1e-6
10	8.65	1e-6
100	1.01	0.012
1000	1.34	0.0008

Table 3: NLL and BLEU scores with varying maximum temperatures for transformer based generator model

Model	NLL_{gen}	BLEU-4
cond-LSTM	3.76	0.003
uncond-LSTM	3.54	0.0004
cond-Transformer	0.98	0.0008
uncond-Transformer	1.08	0.012

Table 4: Conditional vs Unconditional generation

from 1 to 100 over the number of iterations N. We observed similar results when we followed their training strategy. They opted to use 5 discriminator optimization steps for each generator step to account for the difference in power. We observed that this strategy worked best for us as well.

However our goal was slightly different from theirs and is more focused on creating a single training pipeline in an adversarial setting to train the generator to generate good sentences. One additional goal is to use the discriminator as an evaluation metric for comparison as it would know better than any handmade metric how to differentiate two sentences using important and discriminative features.

We find that training a GAN from scratch is very tricky. The generator often goes to a mode collapse and hence the models never converge. We employed tricks like label smoothing and noisy labels and that prevented the discriminator getting too good ($\sim$ 0 disc loss) and never recovering. After some analysis we found that the generator needed more steps to learn in order to fool the discriminator. Hence we tried to explore different step sizes for training the generator. A step size of 2 was convenient and the model seemed to converge at times. However a step size of 4 or higher lead to quick convergence with the generator overpowering the discriminator early in training and never losing again. This is because the discriminator tends to be undertrained when compared to the generator and hence it doesn't learn to produce good sentences. Table 2 denotes the metrics for the transformer model when trained with a cnn discriminator

On experimenting with temperature, we found that high temperatures are necessary for the convergence of the model. We found that the discriminator gets really good at differentiating the real sentences and generated sentences after a few epochs. The generator learns to overcome the discriminator once the temperature reaches 30-50 depending on the model architecture used. This behaviour is consistent with the findings of the Relgan (Nie et al., 2018) paper which claims that the generator first tries to predict sentences of high quality to fool the discriminator. However as the discriminator gets better, the generator has to resort to producing diverse sentences to fool the discriminator. Table 3 denotes this behaviour for the transformer based generator model.

Training tends to be much more stable when we use transformers in comparison to LSTM based models. Although transformer models tend to take longer to converge, the training is more stable and we achieve much better scores in comparison. The difference is especially pronounced with conditional generation owing to the capacity of cross attention using grid features in comparison to the capacity of a flattened image feature representation as shown in Table 4.

Since our goal was to use the discriminator as an evaluation metric after training, we attempted to use a transformer model as the discriminator. However the transformer discriminator had a detrimental effect when used with the LSTM generator. However with the transformer generator, the generator became much worse than the baseline as shown in Table 5. On top of this, to try providing stronger signals for the generator to learn from, we experimented with an autoregressive discriminator which assigns a value for each sub sequence in the sentence as each token gets generated. Since the transformer discriminator did not work well with an LSTM generator, we only tried to use this loss for the transformer based generator

We believe that this behaviour is due to the fact

Model	NLL_{gen}	BLEU-4
LSTM-Transformer	6.54	1e-5
Transformer-Transformer	1.01	0.012
Transformer-ATransformer	2.56	1e-4

Table 5: Transformer as Discriminator

Method	Human Score
Real	4.445
Generated	2.048
MLE	4.013
Discriminator Score	**3.76**

Table 6: Human Evaluation vs discriminator

that the discriminator now becomes too powerful and the generator has a hard time overpowering it. This attributes to the huge difference in power when we use this novel objective function. This function was inspired from the discounted cumulative loss used in Reinforcement Learning to balance the rewards gained in the short term vs in the long term. However we did not experiment with $\gamma < 1$ due to time constraints. We believe that the value of γ may be key to getting a better objective function to optimize.

Finally to compare the performance of the discriminator, we took about 100 images at random and generated scores from the discriminator and scaled it to 1-5 to align with the human evaluation scores that we assigned for the generated captions.

We find that the discriminator score tends to be less strict when compared to human evaluation. Although the scores seem far apart, this is a good way to find out how close the discriminator is to a human evaluator. We hypothesize that the training stability of the model and the strength of the objective used for training may be the reasons for this difference in behaviour. We hope that this encourages future work to compare the behaviour of the discriminator to the human evaluator and analyse where the discriminator attends to during evaluation (in case of a transformer) to analyse the difference in behaviour. On another note, the way we evaluated may be biased and may not be truly aligned with standard human evaluator in Amazon Turks and the rules they had to follow to assign scores to captions.

5 Conclusion and Future Work

In this work we have explored training a Generator for image captioning in an adversarial fashion to generate good sentences and a Discriminator which will double up as an evaluation metric. We found that pre-training the generator played a significant role in training the GAN to convergence which could not be achieved easily with just adversarial training. Mode collapse in GANs is very prominent in case of text generation which makes it difficult for models to converge. Through our experimentations to stabilise training, we found that having additional optimization steps for the generator helped in convergence at times. However, it would be interesting to investigate how MLE based co-training with adversarial training helps address the stability issues. One can also use real tokens to generate the sample and keep the optimization objective consistent during training. We believe this would help in faster convergence.

One future direction is to improve the autoregressive loss function using a discounted cumulative cost to provide information about subsequent generations. Stronger objective functions would allow the generator to have a good enough expressive power while the discriminator would acquire good discriminative power. These discriminators can inturn be used to evaluate the quality of the generated text as they are inherently trained to capture important relationships for identifying good generated text.

References

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*.

Liqun Chen, Shuyang Dai, Chenyang Tao, Haichao Zhang, Zhe Gan, Dinghan Shen, Yizhe Zhang, Guoyin Wang, Ruiyi Zhang, and Lawrence Carin. 2018. Adversarial text generation via feature-mover's distance. In *Advances in Neural Information Processing Systems*, pages 4666–4677.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Matt J Kusner and José Miguel Hernández-Lobato. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. *Advances in neural information processing systems*, 30:3155–3165.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Weili Nie, Nina Narodytska, and Ankit Patel. 2018. Relgan: Relational generative adversarial networks for text generation. In *International conference on learning representations*.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*.

Kaisheng Yao, Trevor Cohn, Katerina Vylomova, Kevin Duh, and Chris Dyer. 2015. Depth-gated lstm. *arXiv preprint arXiv:1508.03790*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-first AAAI conference on artificial intelligence*.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*.